YOLOv8+11-OBB with Alternative Loss Functions and Editable Source Code

Derived from the latest (July 2025) Ultralytics' YOLO, this custom Yolov8+11-OBB repo allows you to clone and edit the source code to accommodate custom loss functions.

To run this model along with the real-time editable source code, there are two Jupyter notebooks: 1. Training_Validation.ipynb, & 2. Batchdetection.ipynb.

Links

- Google Docs: https://docs.google.com/document/d/e/2PACX-1vQUAT_ui45MgeQDcBCJXXyIG5c28rUqNHTVYih-ScA8FMnk_GvEKW02Vi3MLv_cS1UEIM8zUpbBNXpF/pub

Training & Validation

To train YOLO models against a custom dataset & custom loss function, you need to run the "Training_Validation.ipynb" file.

```
!pip install supervision
#in case zip is not found in the shell environment
!sudo apt update
!sudo apt install zip
```

Run this block twice: once before saving the pickle file "datacar.pkl", and once afterwards.

```
j = 0 #switch j between 0 & 1 to run this block twice
for i in range (j,2): #switch j between 0 & 1: before & after saving pickle
    if i == 0:
        !pip uninstall ultralytics -y
        !git clone https://github.com/Suppersine/yolo_alternative_loss_functions.git
    %cd yolo_alternative_loss_functions
    !pip install -e .
   # ~/ depends on the home directory

    if i == 0:
        import os
        os.kill(os.getpid(), 9)  # This will restart the runtime
    else:
        import sys
        sys.path.insert(0, '~/yolo_alternative_loss_functions')

        # Now you can import ultralytics modules
        import ultralytics
        from ultralytics import YOLO
        print(ultralytics.__file__)  # Should point to your cloned directory
```

Once done, run this block to select a loss function, then run the looped block once more.

```
#Extracted from picklesave.py, always run this first in the 2nd iteration
(i=1) in the next block below, because the loss.py file needs to load the
saved .pkl data.
%cd yolo_alternative_loss_functions
import pickle

# List of available modes
modes = ["GBB", "CSL", "KLD_none", "KLD_sqrt", "KLD_ln", "KLD_exp",
"KLD_neg_exp", "KFIOU_dflt", "KFIOU_ln", "KFIOU_exp"]

# Default mode
mode = 'GBB'

print("Select a loss/IoU mode:")
for i, m in enumerate(modes):
    print(f"{i+1}. {m}")

try:
    user_input = input(f"Enter the number corresponding to your choice
(default: {mode}): ")

    if user_input.strip(): # Check if the input is not empty
        selected_index = int(user_input) - 1 # Convert to 0-based index
        if 0 <= selected_index < len(modes):
            mode = modes[selected_index]
        else:
            print(f"Invalid number. Sticking with default mode: {mode}")
    else:
        print(f"No input provided. Sticking with default mode: {mode}")

except ValueError:
    print(f"Invalid input. Please enter a number. Sticking with default mode:
{mode}")

print(f"{mode} loss/IoU mode selected")

# Assign the selected mode (as a string) to myvar
myvar = mode

with open("datacar.pkl", "wb") as f:
    pickle.dump(myvar, f)

print("Variable saved to datacar.pkl")
```

Once done looping, initialise the YOLO model, addresses, essential settings, etc.

```
#check ultralytics address
import ultralytics
from ultralytics import YOLO
print(ultralytics.__file__)  # Should point to your cloned directory
#once the address checking is done, start training the model with (y)our
custom dataset
!jupyter --paths

filedir = '~/obb_dataset/'
yamldir = '~/obb_dataset/datav11.yaml'
homedir = '~/'
yolodir = '~/yolo_alternative_loss_functions/'

#!pip install zip
#!sudo apt-get update && apt-get install -y zip
```

Then, we can commence training the model.

```
# Load a model (N-submodel)
model = YOLO("yolov8n-obb.pt")  # load a pretrained model (recommended for
training)
!yolo task=obb mode=train model=yolov8n-obb.pt data={yamldir} epochs=300
imgsz=640 plots=True
```

Finally, to end the “Training_Validation.ipynb” file & save weights, we will run the validation block. Adjust the nmod integer to count how many submodels you have trained so far.

```
#validation & test
nmod = 4
for i in range (nmod): # where nmod is the number of models trained
    if i == 0:
        weightaddress = str(f'{yolodir}runs/obb/train/weights/best.pt')
    else:
        weightaddress = str(f'{yolodir}runs/obb/train{i+1}/weights/best.pt')
    model = YOLO(weightaddress)
    val = model.val(data=yamldir)

    print("Overall mAPs")

    print(val.box.map) #50/95
    print(val.box.map50) #50
    print(val.box.map75) #75

    print("classwise mAPs")
    print(val.box.maps)
!zip -r yolo_obb.zip runs
```

Batch Detection Test

Check environmental readiness:

```
#in case zip is not found in the shell environment
!sudo apt update
!sudo apt install zip
!pip install supervision
```

Unless there are already pretrained weights in the HDD, run this block to unzip uploaded weights, which must be uploaded to {HOMEDIR}/weights/.

```
%cd weights/
!unzip "*.zip"
%cd ~/ #move to home page
```

Then, check the Ultralytics repo directory.

```
#check ultralytics address
import ultralytics
from ultralytics import YOLO
print(ultralytics.__file__)  # Should point to your cloned directory
```

If absent, rerun the looped blocks, which are structurally identical to those of the training file. You can find their notebook blocks on the 1st and the 2nd pages of this document. Next up, we will define essential paths for the batch detection step.

```
# Once the address checking is done, start training the model with (y)our
custom dataset
!jupyter --paths

filedir = '~/cardiacyv8/'
yamldir = '~/cardiacyv8/datav11.yaml'
homedir = '~/'
weightdir = '~/weights/' # adaptable or…
# weightdir = yolodir + 'runs/'
datadir = '~/obb_dataset/test/images/'
yolodir = '~/yolo_alternative_loss_functions/'

#!pip install zip
#!sudo apt-get update && apt-get install -y zip
```

Using a GUI file explorer (in Jupyter or COLAB's left panel), make sure you count the total number of subfolders within {weightdir} to plan loop nesting.

```
for i in range(4): #where i = number of subfolders in weightdir regardless of
#depth
    !mkdir "results{i+1}"
```

```
#batch detection test
import os
from pathlib import Path  # Improved file path handling
import random
#from ultralytics import YOLO
import supervision as sv
import cv2

k = 0
for i in range(1): #where i is the number of child subfolders
    for j in range(4): #where j is the number of grandchild subfolders (all
must be equal across all i)
        k += 1
        if j == 0:
            weightaddress = str(f'{weightdir}runs/obb/train/weights/best.pt')
        else:
            weightaddress =
str(f'{weightdir}runs/obb/train{j+1}/weights/best.pt')
        model = YOLO(weightaddress)

        # Define the test image directory
        test_dir = Path(f"{datadir}")  # Use Path for better handling

        # Loop through all image files
        for image_path in test_dir.glob("*.png"):  # Modify extension if
needed (e.g., *.png)
            # Perform detection
            results = model(str(image_path))  # Convert Path to string for
YOLO


            # Extract filename
            filename = image_path.name

            # Process and annotate detections (same as your code)
            detections = sv.Detections.from_ultralytics(results[0])
            oriented_box_annotator = sv.OrientedBoxAnnotator()
            annotated_frame = oriented_box_annotator.annotate(
                scene=cv2.imread(str(image_path)),
                detections=detections
            )

            # Define output path (modify as needed)
            output_path = Path(f"{yolodir}/results{k}/{filename}")

            # Save the annotated image (replace with desired format if
needed)
            cv2.imwrite(str(output_path), annotated_frame)
            # Optional: Print progress
            print(f"Detection results saved for: {filename}")
```

Finally, we will zip the detection subfolders to save on your HDD. So that you can view the results on your machine/device.

```
# Create a list of the directory names
%cd ~/yolo_alternative_loss_functions/
results_dirs = [f"{yolodir}results{i}" for i in range(1, 33)]
# where i is the number of result folders we have created before.
# Join the directory names into a single string, separated by spaces
dirs_to_zip = " ".join(results_dirs)

# Construct the full zip command
zip_command = f"zip -r yolov11_detections.zip {dirs_to_zip}"

# Execute the command using Jupyter's shell magic
!{zip_command}

print(f"Created yolov11_detections.zip containing: {dirs_to_zip}")
```